Building an XML document warehouse

Jamel Feki, Ines Ben Messaoud, Gilles Zurfluh

Building an XML document warehouse

Jamel Feki[a]*, Ines Ben Messaoud[a] and Gilles Zurfluh[b]

[a]*Laboratory Mir@cl, University of Sfax, Airport road km 4, P.O. Box 1088, 3018 Sfax, Tunisia;*
[b]*Laboratory IRIT, University of Toulouse 1, Toulouse, France*

Data Warehouses and OLAP (On Line Analytical Processing) technologies are dedicated to analyzing structured data issued from organizations' OLTP (On Line Transaction Processing) systems. Furthermore, in order to enhance their decision support systems, these organizations need to explore XML (eXtensible Markup Language) documents as an additional and important source of unstructured data. In this context, this paper addresses the warehousing of document-centric XML documents. More specifically, we propose a two-method approach to build Document Warehouse conceptual schemas. The first method is for the unification of XML document structures; it aims to elaborate a global and generic view for a set of XML documents belonging to the same domain. The second method is for designing multidimensional galaxy schemas for Document Warehouses.

Les entrepôts de données et les technologies d'analyses en ligne OLAP («On Line Analytical Processing») sont dédiés à l'analyse des données structurées issues des systèmes OLTP («On Line Transaction Processing») des organisations. De plus, ces organisations ont besoin d'explorer des documents XML, comme une importante source additionnelle de données non structurées, à des fins de prise de décisions. Dans ce contexte, cet article s'intéresse à l'entreposage de documents XML orienté-document. Plus particulièrement, nous proposons une approche composées de deux méthodes pour la construction d'un schéma conceptuel d'un Entrepôt de Documents. La première méthode est pour l'unification des structures de documents XML ; elle vise à élaborer une vue globale et générique pour un ensemble de documents XML appartenant à un même domaine. La seconde méthode est pour la modélisation multidimensionnelle en galaxie de schémas d'Entrepôts de Documents.

Keywords: *Document Warehouse*; unification; *XML document*; *multidimensional modeling*; *galaxy schema*

Mots clés: Entrepôt de documents; unification; document XML; modélisation multidimensionnelle; schéma en galaxie

1. Introduction

Textual data represent a means to express information and knowledge. For instance, emails, credit reports and industry newsletters are saved in documents. Actually, documents contain pertinent data for the decision-making process and, therefore, help analysts (i.e., decision makers) to better understand the business processes of their organizations. Indeed, Inmon (2002) emphasizes the importance of external contextual information to understand the results

*Corresponding author. Email: Jamel.feki@fsegs.rnu.tn

of the historical analyses of operations. However, in practice the external contextual information emerges as documents. Furthermore, 80% of information is hidden in documents as non-numeric data (Tseng et al., 2006).

Since the number of documents is greatly increasing, decision-makers spend much of their working time scanning documents and looking for business indicators that help them in their decision-making processes. Obviously, one cannot scan the contents of documents either easily or rapidly. Consequently, during the decisional process, some pertinent documents may be involuntarily missed and, contrariwise, non-pertinent documents may be considered by intuition. The final result may be ineffective since it is based on incomplete or even noisy information (Tseng et al., 2006). In order to correctly exploit these documents which are seen as a vital source for decision analyses, several researchers advocate that documents should be warehoused (McCabe et al., 2000; Sullivan, 2001); thus, the concept of a *Document Warehouse* was born.

Indeed, a Document Warehouse (DocW) is designed to store documents issued from internal and external sources, and then should be organized for effective analyses in order to enable distilled and fruitful business intelligence (Tseng et al., 2006). However, these documents may have different formats. Among these formats, XML (eXtensible Markup Language) is very popular for data representation and exchange on the Web.[1] Thus, in this work, we are interested in *document-centric XML documents*. Usually, these documents have heterogeneous structures; hence, when a decision-maker needs to access them (s)he has to write several queries – as many queries as different structures. To alleviate this querying difficulty, we suggest a two-method approach to build a DocW, namely: (1) *Unification of XML documents structures* (Ben Messaoud et al., 2011a, 2012), to build a global view describing a large set of heterogeneous documents (structural heterogeneity); and (2) *Multidimensional modeling of unified structures of documents* (Ben Messaoud et al., 2011b).

This paper presents these two methods. It is organized as follows: In Section 2, we overview pertinent works related to the unification of XML document structures and to multidimensional modeling of documents. Then, in Section 3, we describe our approach to build an XML document warehouse. In Section 4 we present our unification method for XML document structures. Section 5 describes our multidimensional modeling method. Finally, Section 6 concludes the paper and suggests future work.

2. Related works

The content of an XML document is encapsulated between tags that describe the hierarchical structure of the document. There are two types of XML documents: *data-centric documents* and *document-centric documents*. A data-centric document is highly structured (e.g., sale orders); it is often used to exchange numeric data between applications. On the other hand, a document-centric document contains text (e.g., scientific articles, emails, books) and is less structured. There are two formalisms to describe the structure of XML documents: *DTD* (document type definition) and *XSD* (XML structure definition). They may differ even for documents belonging to the same application domain. We are particularly interested in XML document-centric documents belonging to the same domain.

2.1. Related works dealing with unification

Unification is intended to produce a global view for a set of XML documents structures (i.e., DTD or XSD). In the literature, some works proposed methods to unify DTDs (such as Lee et al., 2002; Mello et al., 2002; Yoo et al., 2005), while others (Zhang et al., 2002) unify

XSDs. In addition, the contribution in Júnior et al. (2008) is a process for XML instance integration.

In Lee et al. (2002) an integration strategy called *XClust* was proposed; it is based on clustering DTDs of XML data sources. In this strategy, similarity degrees between DTDs are computed. These degrees are based on linguistic and structural information, and additionally they consider semantics. In fact, the authors use what they call an 'expansion table' to handle acronyms, and use Wordnet[2] to find synonyms of names. After that, clusters of similar DTDs are determined based on similarity degrees. Finally, an integrated DTD is generated for each cluster. Nevertheless, the similarity calculation depends on the correctness of a tuning phase devoted to set weights and thresholds (De-Meo et al., 2003).

On the other hand, Mello et al. (2002) have proposed a unification method where inputs are conceptual abstractions of DTDs called object-oriented-based canonical schemas. However, this approach requires translating DTDs into the canonical schema. Nevertheless, it does not treat the acronyms present in the DTDs.

In Yoo et al. (2005), the authors propose an algorithm for DTD unification. This algorithm accepts a set of DTDs for XML documents having similar structures and belonging to the same domain; its output is a unified DTD. In fact, the resulting DTD plays the role of a global conceptual schema for subjects common to a given domain. This algorithm includes four steps: *pre-processing, DTD representation, uniform DTD generation*, and *post-processing*. However, in the pre-processing step, the authors use an 'Element Name Resolution Table'; it is used to replace all synonyms with a common and unique term. In practice this table may be incomplete since it cannot cover all synonyms. This step can be improved by the use of domain ontology.

In order to integrate XSDs, Zhang et al. (2002) developed a process that receives a set of XSDs and then generates a global conceptual model. First, it converts each XSD into an *extended UML class diagram (EUML)* and then applies three steps: *clustering of concepts, unification of concepts*, and *restructuring of relationships*. The clustering of concepts resolves the naming conflict using Wordnet. Secondly, the unification step resolves data type and structural conflicts. Finally, the third step restructures relationships and removes redundant ones. The quality of the result model may be low if the input diagrams belong to more than one domain.

A more recent work (Júnior et al., 2008) suggests an integration process for XML instances belonging to the same domain. This process consists of two steps: *similarity definition* and *unification*. First, the similarity definition stipulates a similarity score for each pair of documents. This score calculation is based on metrics and a dictionary for checking synonym terms. This step generates several sets of semantically similar XML documents. Second, the unification generates a unified XML file for each set by using domain ontology and a dictionary. However, the authors propose one binary operator called *Merge* for the unification. This operator provides the union of the XML instances' elements. But, in some cases, the *Merge* operator alone is not enough for the unification step; indeed, *Merge* cannot unify two similar XML instances having different parent elements.

So far, this section has overviewed pertinent works dealing with the unification of XML structures. We emphasize that, in the literature, some works translate XML structures into an appropriate model (tree, EUML diagram, etc.). Other works propose a similarity degree in order to measure the pertinence of integrating XML structures, and consequently to decide which structures can be merged together. Generally, we note that decisional users do not participate during the unification. In Section 4, we suggest a method to unify structures of XML documents. This method is based on an optimized similarity degree calculation relying on a

matrix. In addition, it is interactive, iterative and involves the designer in order to validate the unification result.

2.2. Related works addressing the multidimensional modeling of documents

The multidimensional modeling technology aims to design multidimensional schemas that reflect On Line Analytical Processing (OLAP) requirements. In the literature, there are works regarding the multidimensional modeling of documents. For instance, McCabe et al. (2000) and Tseng et al. (2006) model the DocW as a *star schema* whereas Pujolle et al. (2011) produce a *galaxy schema*.

In McCabe et al. (2000), the authors present a method for searching in text collections. They model the global view of the document set as a multidimensional schema. In fact, they employ the star schema in order to analyze documents. Their star schema distinguishes five types of dimensions, namely *localization, time*, *term, document* and *category*. Its measure (i.e., information to be analyzed) is the number of term occurrences within documents.

Tseng et al. (2006) also use the star schema to analyze documents. This schema distinguishes three types of dimensions: *ordinary* (an ordinary dimension contains keywords extracted from the document), *metadata* (describing the document, e.g., title, author) and *category* (external data for the document description as issued from Wordnet). The star schema measures the number of document occurrences according to dimensions.

Note that star schemas obtained by McCabe et al. (2000) and Tseng et al. (2006) have two main drawbacks: First, they allow only quantitative analyses since their measures are numeric, and second, their subject of analyses (i.e., fact) is defined a priori.

Pujolle et al. (2011) propose a hybrid design process to build OLAP systems from document-centric XML documents. As hybrid, their process combines a top-down approach (starting from user requirements) and a bottom-up approach (relying on the source data model). More interestingly, they suggest a multidimensional conceptual model called a *galaxy schema*. A galaxy schema is uniquely based on the *dimension* concept; it connects several dimensions by nodes instead of facts. A connecting node denotes compatible dimensions for analysis. Nevertheless, the authors do not define rules in order to assist the designer to model XML documents as galaxy schemas.

To recapitulate, we note that we have focused on the related work on multidimensional models: *star* and *galaxy schemas*. In a star schema, the fact represents a predefined subject of analyses. However, in a galaxy the fact is not predefined; it will be specified when querying the schema (among one of the galaxy dimensions); this ensures a certain flexibility/freedom of analyses. Moreover, the galaxy has an additional advantage: It is simpler than the star schema since it is based on a unique concept. Consequently, the designer has no obligation to decide what thing plays the role of dimension, and what is the fact. Considering these benefits, we are interested in the galaxy schema for modeling the DocW (cf. Section 5).

3. Proposed approach for building an XML Document Warehouse

Let's remember that documents to be warehoused may have heterogeneous formats. Among those formats, we are interested in XML since it has been adopted as a de facto standard for exchanged data and, therefore, it is one of the most popular formats in use.

We propose an approach to build the DocW schema. This approach is composed of two methods, namely: (1) *unification of XML document structures*, and (2) *multidimensional modeling* of documents (cf. Figure 1).

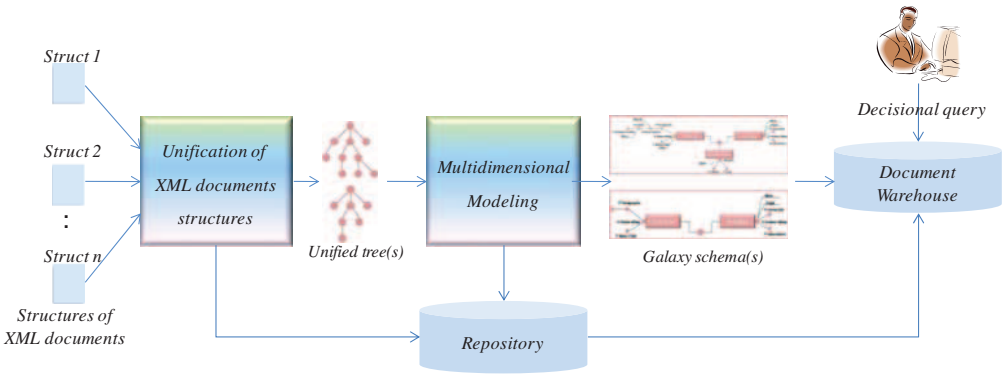


Figure 1. Building an XML (eXtensible Markup Language) Document Warehouse.

The unification method is fundamental since XML documents are often described by heterogeneous structures. It translates a set of input structures into trees by applying two rules (cf. Section 4.1), and then produces a limited number of *unified tree(s)*. In fact, a unified tree plays the role of a global view describing the input document set. The input structures together with the output of this method are saved into a repository to be used in further operations (e.g., mapping, querying). Figure 2a shows the class diagram of this repository.

The multidimensional modeling method produces galaxy schema(s) by applying a set of ten rules (cf. Section 5.2) on trees resulting from the previous step. The galaxy highlights the OLAP components: axes (i.e., dimensions) and perspectives (i.e., hierarchies). The galaxy schemas are saved in the repository described by the class diagram of Figure 2b.

In the following, we detail the first method in Section 4 (i.e., unification of XML document structures), and the second method (i.e., multidimensional modeling) in Section 5.

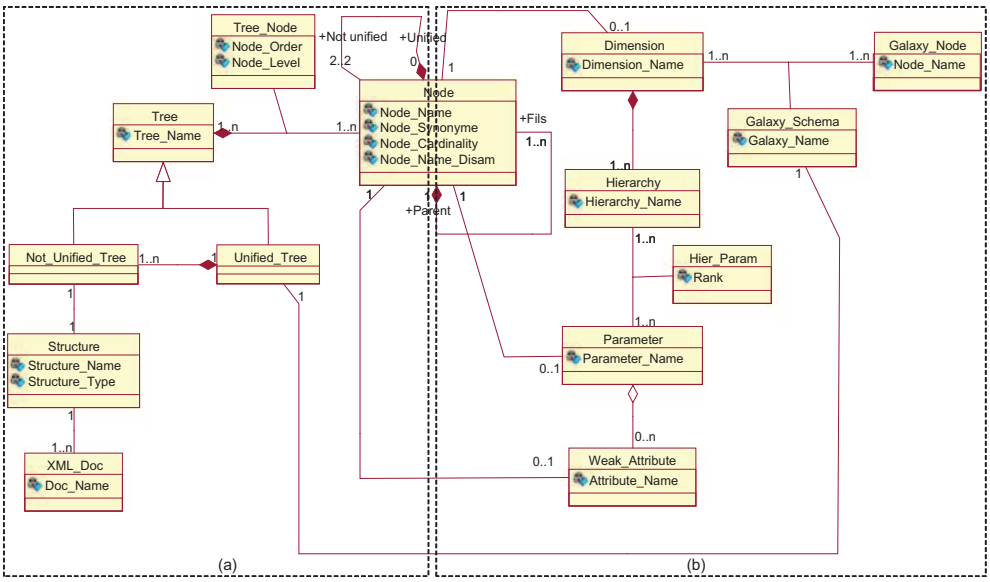


Figure 2. Class diagram of the Document Warehouse repository.

4. Unification of XML document structures

In general, XML document structures differ even if these documents belong to the same activity domain. Consequently, when a decision-maker needs to query these documents, (s)he has to consider their structural heterogeneity, and therefore should write several queries – as many queries as the number of distinct structures. After that, (s)he needs to gather the returned query results to obtain the final result. In fact, this represents a technical and tedious task. To alleviate this problem, we expect to define unified tree(s) that play(s) the role of a general view for a whole set of documents.

In Ben Messaoud et al. (2011a, 2012), we proposed a method for unification of XML documents belonging to the same domain. This method exploits the tree structure of XML documents and is based on a similarity factor calculation. Figure 3 depicts the four main steps of this method, namely: (1) *tree representation*, (2) *generation of unified trees*, (3) *validation of unified trees*, and (4) *correctness verification of trees*. The remainder of this section presents these five steps.

4.1. Tree representation

This step translates the input structure of each XML document into a graphical representation; we elected to use the formalism of the tree as done in Yoo et al. (2005) and Lee et al. (2002) because it is more convenient to unskilled persons who may be involved. We perform this translation according to the following two rules:

Rule 1: Each element or attribute of the XML structure constitutes a tree node annotated with cardinality derived from XML tags.

Rule 2: Each pair of XML elements defined within the same tag transforms into an arc that indicates the relationship between this pair of elements.

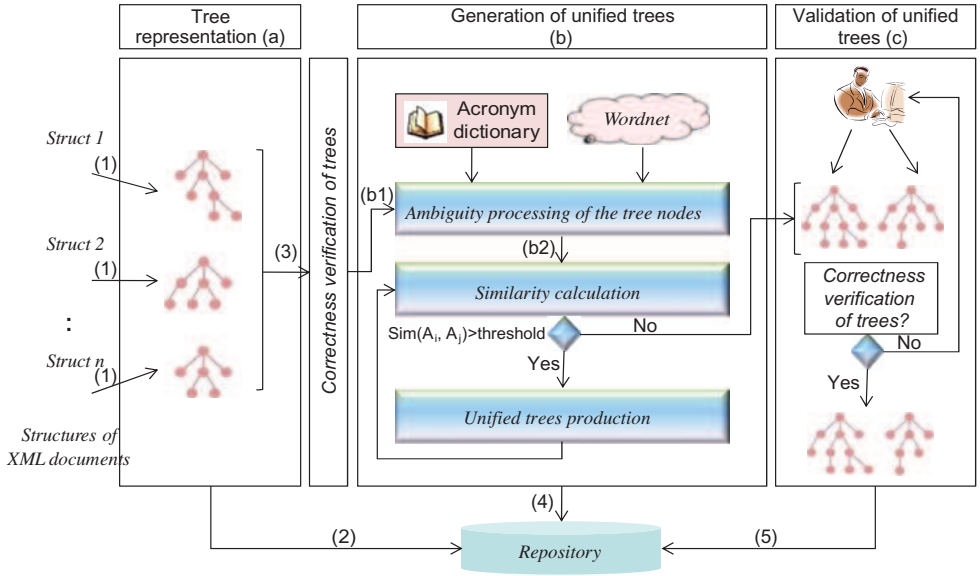


Figure 3. Unification of XML (eXtensible Markup Language) documents structures.

We define a tree as follows:

Definition 1: A tree T is defined by the triplet *(E, r, N)* where
- $E = \{e_1, e_2 ..., e_n\}$: is a non empty set of n nodes.
- $r \in E$: is the root node of tree T.
- $N = \{n_1, n_2 ..., n_k\}$: is the set of k arcs of T.

Each node is characterized by a name and cardinality.

Definition 2: A node $e_i \in E$ is defined by the couple *(name, card)* where
- *name*: is the node name.
- *card*: is the cardinality of the node.

We adopt the tree representation since it is easily understood by decision-makers; this motivates them later to participate in the validation step.

4.2. Generation of unified trees

The generation of unified trees accepts a set of trees, applies a set of operators and then generates unified tree(s) (Ben Messaoud et al., 2011a, 2012). It is composed of the three following sub-steps (cf. Figure 3.b) called:

– Ambiguity processing of the tree nodes,
– Similarity calculation, and
– Unified trees production.

4.2.1. Ambiguity processing of the tree nodes

The objective here is to resolve ambiguities of node names. To do so, we build an *acronym dictionary* and we use the lexical database *Wordnet*. The *dictionary* provides for an acronym its complete word (e.g. *Author* for *Aut*). We have used the *Jaro* algorithm[3] (Jaro, 1989) in order to build this dictionary. *Wordnet* returns a synonym set for a given word. In fact, nodes having the same meaning are detected and substituted with a standard name which is the most frequently occurring term in the synonym set.

Also, this sub-step ensures the uniqueness of nodes names, so it renames nodes having the same name by prefixing each one with the name of its immediate parent. This is to obtain an accurate similarity factor based on nodes.

4.2.2. Similarity calculation

The similarity calculation receives a set of trees having unique and standard node names and then computes the similarity degree between each pair of trees in order to determine trees that have to be merged. It is based on a triangular *Similarity Matrix* (*SM*) having n trees in rows and in columns. This matrix facilitates the identification of trees to be merged. It is inspired by the multidimensional schema integration matrix used in Feki (2004). Each cell *SM(i,j)* contains the *Similarity factor* $Sim(T_i,T_j)$ between trees in row i and tree in column j. The similarity factor is calculated according to Definition 3.

Definition 3: The similarity factor of two trees T_i (E_i, r_i, N_i) and T_j (E_j, r_j, N_j) is:

$$Sim\ (T_i,\ T_j)\ =\ \begin{cases} 0.75\ if\ n_i = c_{i,j}\ and\ n_i < n_j \\ \frac{c_{i,j}}{q}\ otherwise \end{cases}$$

where:
$n_i = |\ E_i\ |,\ n_j = |\ E_j\ |,\ c_{i,j} = |\ E_i \cap E_j\ |\ and\ q = n_i + n_j - c_{i,j}$

We note that the fusion of two trees T_i *and* T_j is meaningful only when $Sim(T_i, T_j)$ is greater than a given threshold that can be determined by experimentation.

4.2.3. *Unified trees production*

This step produces a merged tree for each pair of trees having their similarity factor higher than the threshold. To perform this merge, we propose three operators: *fusion by inclusion, fusion by union of sub-trees* and *fusion by merging nodes*.

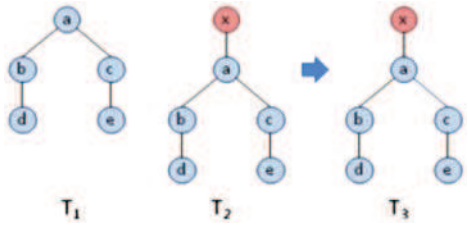
The *fusion by inclusion* of two trees T_1 and T_2 is performed when one of the two trees is included within the other. The resulting tree, denoted T_3, is the greatest among T_1 and T_2. Definition 4 describes the *fusion by inclusion* operator.

Definition 4: ***F-Inclusion*** $(T_1, T_2) = T_3$

Inputs:
- T_1 (E_1, r_1, N_1).
- T_2 (E_2, r_2, N_2).

Condition:
- $T_1 \subseteq T_2$ or $T_2 \subseteq T_1$.

Output:
- $T_3 = T_2$ if $T_1 \subseteq T_2$.
- $T_3 = T_1$ if $T_2 \subseteq T_1$.



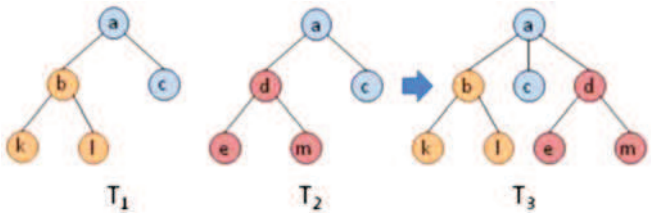
The *fusion by union of sub-trees* operator of two trees T_1 and T_2 is required when two different sub-trees in T_1 and T_2 have the same parent node name in T_1 and T_2. The resulting tree, called T_3, is composed of all sub-trees issued from T_1 and T_2. This operator is formalized in Definition 5.

Definition 5: ***F-Sub-Trees*** $(T_1, T_2) = T_3$

Inputs:
- T_1 (E_1, r_1, N_1).
- T_2 (E_2, r_2, N_2).

Conditions:
- $E_1 \subseteq E_2 \neq \varnothing$.
- $\exists\ e_i \in E_1$ and $e_j \in E_2$, $\forall\ e_i = e_j$, Parent (e_i) = Parent (e_j) and Child (e_i) ≠ Child (e_j).

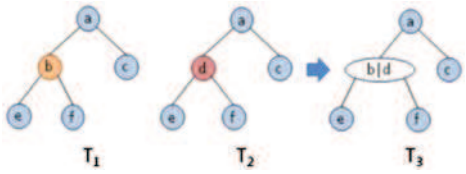
/* Parent (e_i) returns the parent of node e_i */



/* Child (e_i) determines sub-tree of
node e_i */
Output:
- T_3 (E_3, r_3, N_3) with
- $E_3 = E_1 \cup E_2$
- $r_3 = r_1 = r_2$
- $N_3 = N_1 \cup N_2$

The *fusion by merging nodes* operator of two trees T_1 and T_2 is realized when the same sub-trees exist in T_1 and T_2, and those sub-trees have parents with different node names. It produces a new tree T_3 composed of a specific node (i.e., node OR) linked to the common sub-tree. Definition 6 describes this operator.

Definition 6: ***F-Merging-Nodes*** *(T_1, T_2) = T_3*
Inputs:
- T_1 (E_1, r_1, N_1).
- T_2 (E_2, r_2, N_2).
Conditions:
- E_1 $E_2 \neq \varnothing$
- $\exists$ $e_i \in E_1$ and $e_j \in E_2$ $\forall$ $e_i \neq e_j$, Child(e_i) = Child(e_j) and Parent(e_i) = Parent(e_j).
Output:
- T_3 (E_3, r_3, N_3) with
- $E_3 = (E_1 - \{e_i\}) \cup (E_2 - \{e_j\}) \cup (e_i \mid e_j)$ /* ($e_i \mid e_j$) is a specific node OR */
- $r_3 = r_1 = r_2$
- $N_3 = N_1 \cup N_2$



Note that during the production of unified trees, cardinalities are treated according to a set of ten rules adopted from Hachaichi et al. (2010). These rules replace multiple appearances of the same element with one element having the highest cardinality.

Note that we reuse these rules on XSD documents. Indeed, when an element e_i *(name, card)* $\in E$ has a numeric cardinality (*card* > *1*) then we replace this cardinality with (*) in the tree issued from the first step (i.e., Tree representation) and then we apply the rules of Figure 4 on element e_i.

e1*, e1*→ e1*
e1*, e1?→ e1*
e1?, e1*→e1*
e1?, e1?→ e1*
e1, e1→ e1+
e1+, e1+→ e1+
e1+, e1?→ e1+
e1?,e1+→ e1+
e1+,e1*→ e1+
e1*, e1+→ e1+

Figure 4. Rules treating the cardinalities of trees.

4.3. Validation of unified trees

In this step, the resulting trees issued from the second phase (i.e., generation of unified trees) are supplied to the decision-maker/designer for validation according to their analytical requirements; (s)he can delete and/or rename nodes. All changes are saved in the repository of Figure 2.a. This repository is later used in the multidimensional modeling method, and in querying XML documents.

4.4. Correctness verification of trees

In order to build trees that are syntactically correct, we defined in Aouabed et al. (2012) a set of four constraints. They are:

– *Connected nodes*: It requires that each node belonging to a tree has to be connected by at least one arc.

This constraint ensures that each tree does not contain any isolated nodes. Consequently, when the decision-maker expresses analytical requirements, (s)he can use any multidimensional element. We note that in multidimensional modeling (cf. Section 5), each node will be transformed into a multidimensional element.

– *Hierarchy*: It requires that each node is linked to a unique parent node.

This constraint verifies that each node has only one parent node. Thus, we obtain well structured trees.

– *Uniqueness of the root node*: It ensures that each tree has exactly one root node.

This constraint forces building trees with no disconnected sub-trees. It enhances building a correct multidimensional model on the tree: All the multidimensional elements will be connected.

– *Acyclicity*: It checks the absence of cycles within a tree. It requires that a node cannot be parent and child of the same node by transitivity.

The acyclicity constraint allows the generation of multidimensional schemas having no cycles. Hence, it prevents recursive and infinite drill-down/roll-up operations.

4.5. USD tool

To validate our unification approach for XML document structures we have implemented a software tool baptized *USD* (*Unification of Structures of XML Documents*). USD accepts a set of XML structures (i.e., DTD or XSD) issued from the same domain and then generates one or more unified tree(s). Note that in order to generate a DTD/XSD for XML documents not having one, USD invokes *XMLSpy*[4]. In the remainder, we describe the USD functionalities and illustrate them through the unification of a set of 50 XML scientific papers conform to the four DTDs illustrated in Figure 5.

The first step (i.e., tree representation) produces a tree for each input DTD. As an example, Figure 6 shows *Tree1* built on *DTD1*. This tree has 11 nodes, E = {*Article*, *Title*, *Writer*, *Section*, *Day*, *Month*, *Name*, *Affiliation*, *Title*, *Subdivision*, *Para*} where the root is {*Article*}, and a set of 10 arcs N = {(*Article–Title*), (*Article–Writer*), (*Article–Section*), etc.}. In *Appendix 1,* the reader can find *Tree2, Tree3 and Tree4* built respectively on *DTD2* to *DTD4*.

DTD1

<!ELEMENT Article (Title, Writer+, Section+, Day, Month)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Writer (Name, Affiliation)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Affiliation (#PCDATA)>
<!ELEMENT Section (Title?, Subdivision+)>
<!ELEMENT Subdivision (Para)>
<!ELEMENT Para (#PCDATA)>
<!ELEMENT Day (#PCDATA)>
<!ELEMENT Month (#PCDATA)>

DTD2

<!ELEMENT Article (Title, Auth+, Body, References+, Day, Month, Year)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Auth (Name, Institute)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Institute (#PCDATA)>
<!ELEMENT Body (Section+)>
<!ELEMENT Section (Paragraph+)>
<!ELEMENT Paragraph (#PCDATA)>
<!ELEMENT References (#PCDATA)>
<!ELEMENT Day (#PCDATA)>
<!ELEMENT Month (#PCDATA)>
<!ELEMENT Year (#PCDATA)>

DTD3

<!ELEMENT Article (Title, Author+, Outline, Section+, Month, Year)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (Name, University)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT University (#PCDATA)>
<!ELEMENT Outline (#PCDATA)>
<!ELEMENT Section (Section_Number, Title, References, Subsection+)>
<!ELEMENT Section_Number (#PCDATA)>
<!ELEMENT References (#PCDATA)>
<!ELEMENT Subsection (Subsection_Number, Table?, Figure?)>
<!ELEMENT Subsection_Number (#PCDATA)>
<!ELEMENT Table (#PCDATA)>
<!ELEMENT Figure (#PCDATA)>
<!ELEMENT Month (#PCDATA)>
<!ELEMENT Year (#PCDATA)>

DTD 4

<!ELEMENT Article (Tit, Author+, Abstract, Body)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (Name, Affiliation)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Affiliation (#PCDATA)>
<!ELEMENT Abstract (#PCDATA)>
<!ELEMENT Body (#PCDATA)>

Figure 5. Example of four DTDs (document type definitions).

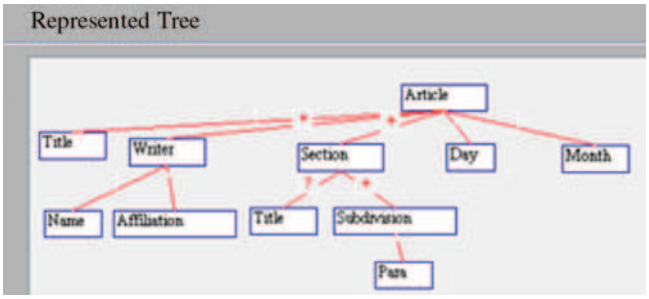


Figure 6. Tree1: Graphical representation for tree of DTD1 (Document Type Definition 1) [(Unification of Structures of XML Documents) (USD) tool].

Second, the ambiguity processing step resolves both synonyms and acronyms. Consequently, in *Tree1* the *Para* node is substituted with *Paragraph;* also, the *Aut* node in *Tree2* is replaced by *Author*, referring to the *Acronym dictionary*. Furthermore, in *Tree1* the *Writer* node is replaced with its synonym *Author* which is much more frequent in the remaining trees. Moreover, the *Subdivision* node in *Tree1* and the *Outline* node in *Tree3* are changed respectively by their synonyms *Subsection* and *Abstract* by using *Wordnet*. In addition, this step treats the ambiguities of tree nodes names; as a result, in *Tree1* and *Tree3* there are two nodes having the same name *Title*, they are renamed *Article_Title* and *Section_Title*.

Third, the similarity matrix is computed in order to find trees to be merged. The first iteration produces the matrix in figure 7.

For our running example, we arbitrarily set the threshold value to 0.4. Since *Sim (Tree1, Tree2) = 0.5* (greater than the threshold) then *Tree1* is merged with *Tree2* using the *Union of sub-trees* operator that produces *Tree*' of Figure 10.

In the second iteration, the similarity calculation produces the subsequent matrix (figure 8).

In this matrix, the highest similarity value *0.55* is greater than the threshold (0.4) then *Tree*' will be merged with *Tree3* through the *union of sub-trees* operator giving *Tree*" (cf. Figure 11).

After the third iteration, the similarity calculation produces the matrix in figure 9.

Since *Sim (Tree", Tree4) = 0.75* is greater than the threshold then *Tree*" will be merged with *Tree4* using the *Fusion by inclusion* operator. Since *Tree4* is included in *Tree*" thus this latter is the final unified tree.

Finally, the result tree is given to the decision-makers/designers in order to adjust it to their analytical requirements.

Threshold: 0.4

Matrix1

	Tree1	Tree2	Tree3	Tree4
Tree1	-	0.5	0.42	0.38
Tree2	-	-	0.45	0.33
Tree3	-	-	-	0.27
Tree4	-	-	-	-

Figure 7. Similarity matrix calculated after the first iteration.

Threshold: 0.4

Matrix2

	Tree'	Tree3	Tree4
Tree'	-	0.55	0.35
Tree3	-	-	0.27
Tree4	-	-	-

Figure 8. Similarity matrix obtained after the second iteration.

Threshold: 0.4

Matrix3

	Tree"	Tree4
Tree"	-	0.75
Tree4	-	-

Figure 9. Similarity matrix: final step.

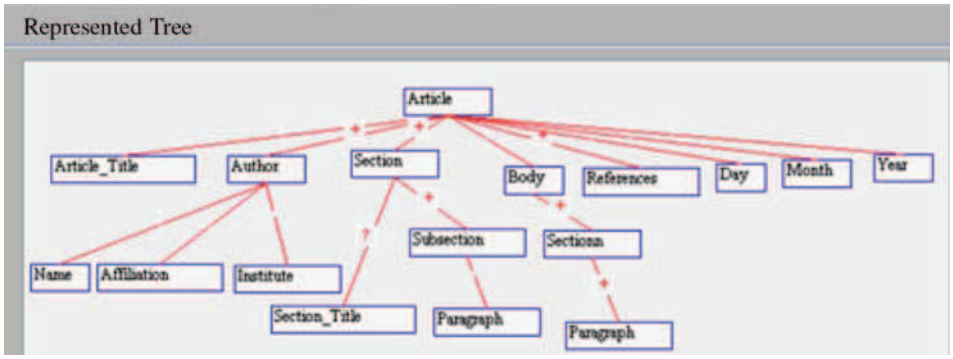


Figure 10. Tree': Tree resulting from the fusion of Tree1 and Tree2.

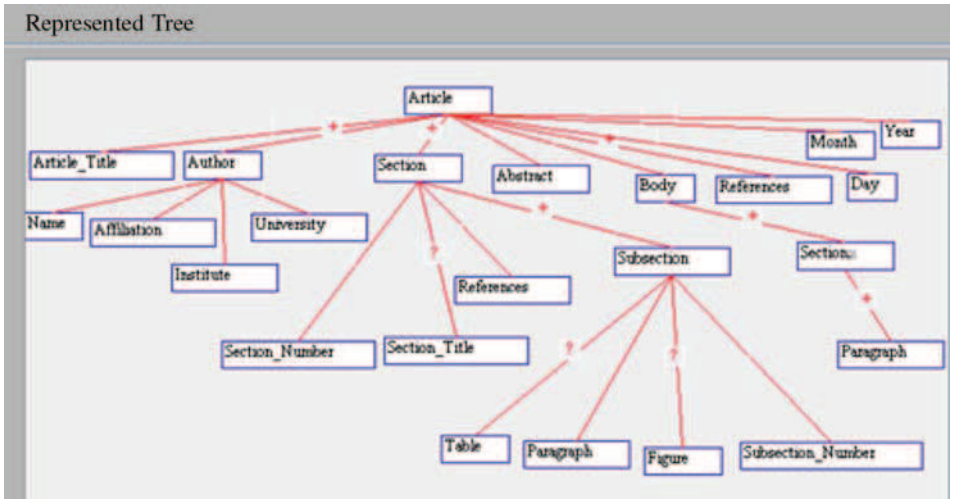


Figure 11. Tree": Final tree resulting from the fusion of Tree' and Tree3.

5. Multidimensional modeling

Multidimensional modeling is for the design of multidimensional schemas supporting OLAP analyses for decisional purposes. In the literature review (cf. Section 2.2.), we have discussed two main multidimensional schemas and opted for the galaxy schema.

In order to build galaxy schemas for XML documents, our method accepts validated/unified trees as built in Section 4.5. It consists of the four following steps: (1) *Pretreatment of trees*, (2) *Building galaxy schemas*, (3) *Galaxy schemas validation*, and (4) *Correctness verification of galaxies*.

Figure 12 depicts these multidimensional modeling steps that we will detail; an example of a galaxy schema is depicted by Figure 13.

The galaxy schema of Figure 13 is composed of four dimensions: *D-Article, D-Date, D-Author* and *D-References*.

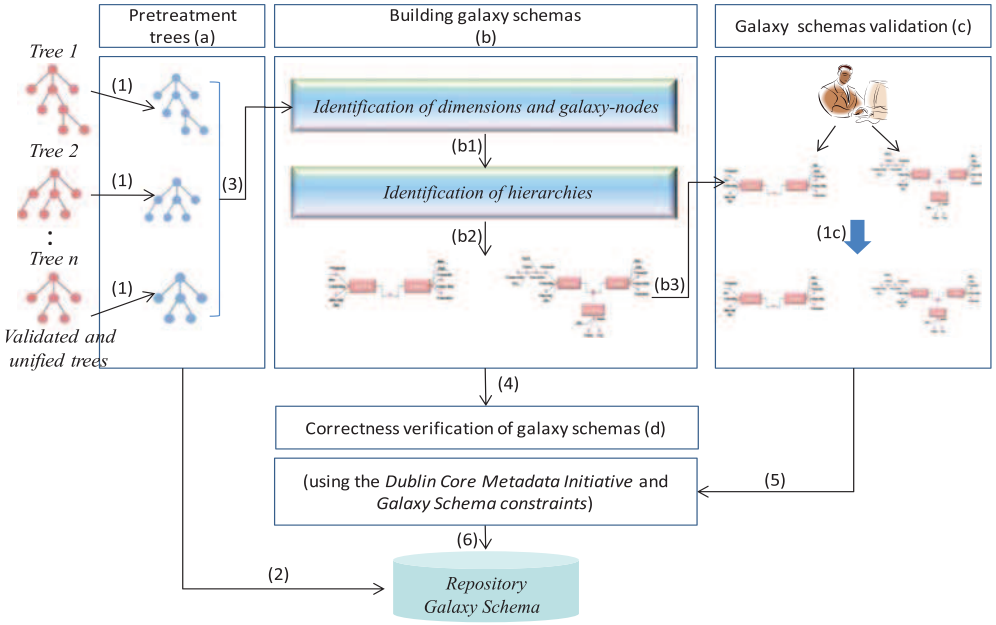


Figure 12. Multidimensional modeling method.

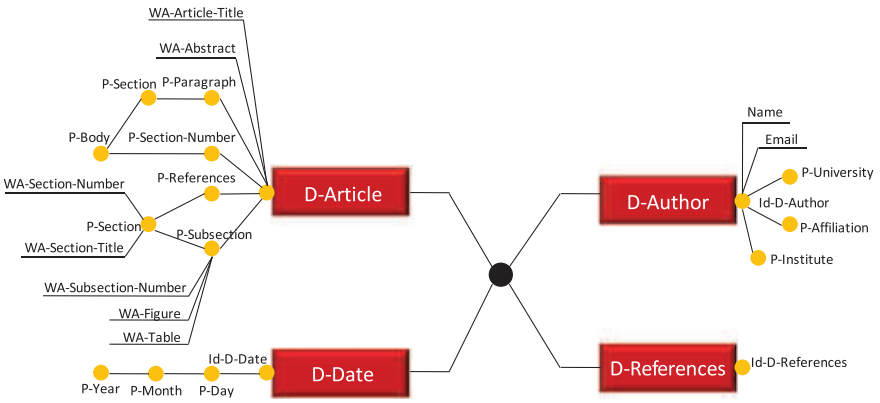


Figure 13. Galaxy schema for the Document Warehouse built on the four DTDs (document type definitions) of Figure 5.

5.1. *Trees pretreatment*

The pretreatment step accepts trees resulting from the first method of our DocW building approach (i.e., unification of XML document structures) and then generates for each tree a pretreated one. In fact, it improves the semantics of the input trees: It automatically adds cardinalities by examining a set of documents compliant to the original DTDs.

5.2. *Building galaxy schemas*

This step transforms each pretreated tree into a galaxy schema by applying a set of 10 rules (Ben Messaoud et al., 2011b). It is conducted in two sub-steps (cf. Figure 12.b):

– Identification of dimensions and galaxy-nodes, and
– Identification of hierarchies.

5.2.1. Identification of dimensions and galaxy-nodes

Within a galaxy schema, a dimension may stand for an analysis axis or even a potential analysis subject. A dimension is described by a name and a set of parameters organized into hierarchies.

5.2.1.1. Dimensions identification. We define the three following rules to determine dimensions from a pretreated tree:

Rd1: The root node *r* of a tree transforms into a dimension called *D-r*.

Obviously, the root node is the most generic node of a tree. Consequently, it can represent only a *dimension-node* (i.e., a node identified as a dimension).

Rd2: Each pair of non-terminal nodes *M* and *N* related by an arc *M-N* annotated with cardinality + or * in the two sides of *M-N* constitutes two dimensions called *D-M* and *D-N*.

The cardinalities + or * in each side denote the presence of a relationship (in the sense of the E/R diagram) between these two nodes. Thus, we consider these two nodes as dimensions.

Rd3: Nodes describing a date (e.g., month, year) and stemming from the same parent node represent a temporal dimension named *D-Date* where these nodes are its parameters.

In a data warehouse, data are considered as chronological series of values; therefore, the temporal dimension is systematically present (Kimball, 1997). Thus, in order to define the temporal dimension, first we detect *Date* nodes and then organize them; this organization may be done referring to a standard *Date* dimension.

Once the dimensions are identified, we follow with the galaxy-nodes identification.

5.2.1.2. Galaxy-nodes identification. Dimensions are related by galaxy-nodes to constitute groups of compatibles dimensions. We define rule *Rn* to determine galaxy-nodes.

Rn: Every pair of nodes *N* and *M*, identified as dimensions, directly related by an arc *N-M* in the pretreated tree constitutes two compatibles dimensions.

Remember that compatible dimensions are related by a galaxy-node. Obviously, since nodes denoting dimensions are related by an arc in the pretreated tree, so they merit being connected in the galaxy schema.

5.2.2. Identification of hierarchies

In multidimensional modeling, a dimension's attributes (i.e., parameters) are organized into one or more hierarchies from the finest towards the highest granularity. In fact, hierarchies represent analysis perspectives for a given dimension (Ravat and Teste, 2000). The lowest granularity level is the identifier of a dimension. In our work, as in data warehousing practice, we define a surrogate key for each dimension produced by rules Rd_1 to Rd_3.

5.2.2.1. Parameter determination. We propose four rules to identify parameters for the generated dimensions. Each parameter is described by its name and its level.

Rp1: Each terminal node *N* having its parent node *M* identified as a dimension and related with an arc *M-N* not annotated with the cardinalities *1-1* becomes a parameter called *P-N* at level 2.

Clearly, a terminal node *N* related to a dimension-node *M* can play only a parameter role. We exclude terminal nodes related with an arc *N-M* annotated with cardinalities *1-1* since these nodes represent descriptive information for the parent node *M*. Descriptive information cannot stand for a parameter.

Rp2: Each terminal node *N* having a parent node *M* identified as a parameter with the level *i* and related with an arc *M-N* not annotated with the cardinalities *1-1*, constitutes a parameter called *P-N* at level (*i-1*).

A terminal node *N* linked to a parameter node *M* where the arc *N-M* is not annotated with the cardinalities *1-1* represents a parameter. Since within a tree every parent node *N* groups detail nodes, consequently the most deep parameter node is the finest parameter level.

Rp3: Every non-terminal node *N* linked to a parent node *M* identified as a parameter node at a level *i* where the arc *M-N* has cardinalities 1-(+ or ∗), becomes a parameter *P-N* for the level *i-1*.

Rp4: Each non-terminal node *N* linked to a parent node *M* identified as a dimension node where the arc *N-M* is not annotated with the cardinalities + or * in the two sides of *M-N* constitutes a terminal parameter.

5.2.2.2. Weak attribute determination. Descriptive data, so-called weak attributes, can be associated with parameters. We define two rules *Rw1* and *Rw2* to extract such weak attributes from a pretreated tree.

Rw1: Every terminal node *N* having its parent node *M* identified as a dimension *D* and the arc *N-M* annotated with the cardinalities *1-1* or *1-0* represents a weak attribute for the identifier of *D*; conventionally this weak attribute is called *W-N*.

Rw2: Every terminal node *N* having its parent node *M* identified as a parameter *P* and the arc *N-M* annotated with the cardinalities *1-1* or *1-0* represents a weak attribute for *P*; we call this weak attribute *W-N*.

A terminal node *N* related to a node *M* through an arc *N-M* annotated with the cardinalities *1-*(*1* or *0*) represents descriptive information for *M*. We consider these nodes as weak attributes.

Applying these rules (defined in Sections 5.2.1 and 5.2.2) to *Tree-E* (cf. Figure 14) of our running example we obtain a galaxy schema. In general, each generated schema will be syntactically verified by checking validation rules (cf. Section 5.3) and then supplied to the designer for validation.

5.3. Galaxy schemas validation

Galaxy schemas resulting from the previous step are validated by decision-makers according to their analytical requirements. They can delete and/or rename multidimensional elements (e.g., dimension, parameter) and reorganize parameters. All changes are saved in the repository of Figure 2b to be used later in querying galaxies.

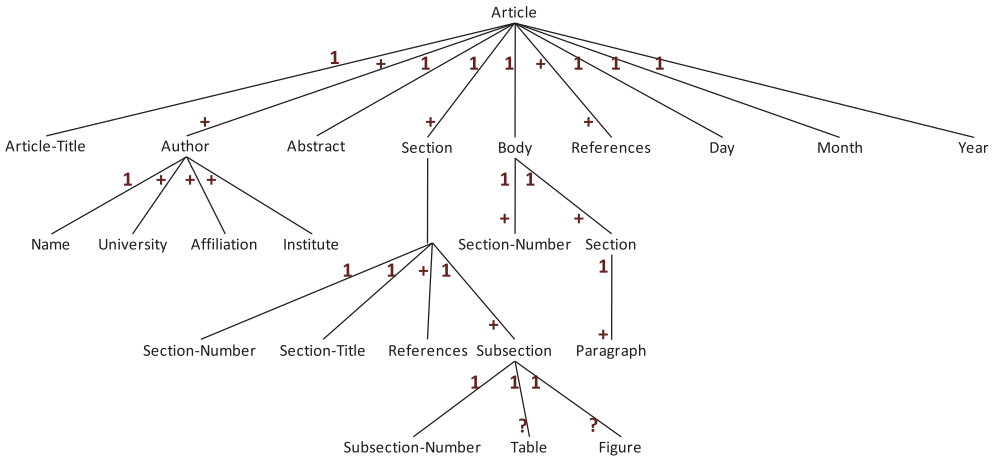


Figure 14. Tree enriched with cardinalities: Tree-E.

5.4. Correctness verification of galaxies

As we aim to build correct schemas, we expect to verify the correctness of the output schemas issued from steps two and three. For this, we define constraints to be verified on the result galaxy schema. Furthermore, we detect that, occasionally, a generated dimension may contain information describing the structure of the document and others representing metadata. To avoid such a conflicting situation, we adopt the *Dublin Core Metadata Initiative*[5] to detect hierarchies describing the metadata of the documents so that we create a new appropriate dimension for these hierarchies.

In the literature, some works define constraints for star and constellation schemas (Ben-Abdallah et al., 2009; Carpani et al., 2001; Ghozzi et al., 2003; Hurtado et al., 2002). However, to the best of our knowledge, no constraints were defined for galaxy schemas. Nevertheless, we note that there are similarities between star and galaxy schemas. Consequently, we have inherited, from the star-schema, eight of its constraints for the galaxy. Additionally, we define three new specifics ones. We classify the set of galaxy schema constraints into three classes, relative to dimensions, nodes or hierarchies.

5.4.1. Dimension constraints

Cd1: *Identification constraint*: Each dimension must have an identifier. It may be either a key from the data source or a surrogate key (Carpani et al., 2001; Hurtado et al., 2002).

Cd2: *Non-empty dimension*: Every dimension should have at least one hierarchy (Ben-Abdallah et al., 2009).

Cd3: *Non-isolated dimension*: In a galaxy schema, every dimension has to be related to n $(n \geqslant 1)$ nodes.

5.4.2. Node constraints

Cn1: *Non-isolated node*: Each node has to be linked to at least two different dimensions. This permits us to perform multidimensional analyses on n ($n \geqslant 2$) axes.

Cn2: *Disjunction of nodes*: There are no direct links between nodes in a galaxy schema. The only links between nodes are indirect via dimensions.

5.4.3. Hierarchy constraints

Ch1: *Hierarchical root*: All hierarchies of a dimension *D* start from the identifier of *D* (Ben-Abdallah et al., 2009).

Ch2: *Exclusive hierarchies*: Any dimension having the minimal hierarchy[6] must not have other hierarchies (Ben-Abdallah et al., 2009).

Ch3: *Non-isolated attribute*: In a dimension *D*, each attribute must belong to at least one hierarchy of *D* (Ben-Abdallah et al., 2009).

Ch4: *Non-empty hierarchy*: Within a dimension *D*, a hierarchy must contain at least two parameters: the identifier of *D* and the *All* parameter (Ben-Abdallah et al., 2009).

Ch5: *Roll-up*: All the parameters of a hierarchy, except the *All* parameter, have at least a parent (Hurtado et al., 2002).

Ch6: *Acyclicity*: Any parameter, except *All*, cannot be parent and child for the same parameter by transitivity (Ghozzi et al., 2003; Hurtado et al., 2002).

5.5. Example

In this section, we apply our multidimensional modeling approach on the unified tree *Tree*'' of Figure 11 (built on DTDs of Figure 5). During the first step *Tree*'' undergoes a pretreatment by adding cardinalities for each parent node. These cardinalities are determined by examining an XML document set conforming to the original DTDs. Figure 14 shows *Tree-E* enriched with cardinalities.

Secondly, the multidimensional elements of the galaxy schema are extracted by applying our set of rules defined in Section 5.2. In our method, we first determine dimensions. As an example, the *D-Article* dimension is built on the *Article* node by applying rules *Rd1* and *Rd2*. In Appendix 2, Table 1 lists for each extracted dimension its source node-name with its extraction rule.

Afterward, we define for each extracted dimension a surrogate key (*Id-D-Article*, *Id-D-Author*…). Then, the inter-dimension nodes are defined. For our running example, one node connects all the four dimensions.

We apply six rules (cf. Section 5.2.2) to determine parameters and weak attributes. In Appendix 3, Table 2 gives, for each dimension, its ordered parameters and its weak attributes. The final result is the galaxy schema of Figure 13.

6. Conclusion and future works

The Data Warehouse technology allows management and analyses of huge amounts of structured data. Nevertheless, this technology did not pay enough attention to documents. Really, documents (e.g., document-centric XML documents) constitute an important source of information for decisional stakeholders. Therefore, they largely merit being warehoused. In this paper, we have proposed an approach to build a Document Warehouse (DocW) schema. This approach is composed of two main semi-automatic methods, namely *unification of XML documents* and *multidimensional modeling*.

The *unification method* accepts a set of XML document structures and then produces unified tree(s). First, it translates each XML document structure into a tree; this step facilitates understanding and visualizing the original XML structure and, therefore, enables unskilled

persons to intervene during the next steps. Second, the ambiguity of tree nodes is solved using Wordnet and a dictionary of acronyms that we have specifically created for that purpose. In order to merge trees, we have defined a similarity factor. It estimates how trees are closely similar; it is evaluated for each pair of trees using a similarity matrix in order to find which trees have to be merged first. For this merge processing, we have defined three algebraic operators: *fusion by inclusion, fusion by union of sub-trees* and *fusion by merging nodes*. Finally, generated tree(s) are submitted to the designer for validation. Furthermore, our unification method verifies the correctness of the generated trees by checking a set of specific constraints.

The *multidimensional modeling method* translates each unified tree issued from the unification into a galaxy schema. To do so, first cardinalities are added for each node. Second, the multidimensional concepts (e.g., dimensions, parameters) are extracted by applying a set of 10 rules, and then galaxy schemas are produced. Finally, the designer can examine and validate the resulting galaxy. In order to generate valid schemas, we have defined constraints at two abstraction levels: four constraints are applied on unified trees and 11 constraints are applied on galaxy schemas.

For assessment purposes, we have developed a software prototype that supports both methods of our approach. Also, we have conducted preliminary evaluation tests and obtained encouraging results. Due to the complete absence of a benchmark for galaxy schemas and their DTD/XSD scripts, our initial tests were feasibility tests performed on 15 DTDs taken from the academic domain. Compared to the three galaxy schemas built manually (on the 15 XML structures), two issued from the prototype are identical, one has one additional dimension, and one galaxy schema has one dimension less, whereas hierarchies are almost the same.

In a further real evaluation test, we conducted experiments on a set of 1691 XML documents taken from the medical collection Clef-2007 described by three DTDs (cf. Appendix 4). However, there were some inadequacies in these documents: for example, *keywords* are gathered inside a unique textual tag. In order to alleviate this difficulty we have improved the initial DTDs (by adding the + cardinality to some elements) to obtain more accurate XML documents.

The evaluation test produces one galaxy schema describing the set of all Clef-2007 documents. Appendix 5 gives two interfaces; the first one is for extracted dimensions (*D_Casimage_Case, D_KeyWords, D_References, D_Reviewer* and *D_Author*); the second interface illustrates extracted hierarchies for the *D_Casimage_case* dimension. The galaxy schema issued from the prototype is identical to the one built manually.

For the near future, we aim to consolidate our tests using a larger number of DTD/XSDs. As a sShort-term perspective, we envisage implementing a query language for the galaxy-based document warehouse.

Notes

1. Extensible Markup Language (XML): http://www.w3.org/xml/.
2. Wordnet: http://wordnet.princeton.edu/.
3. The Jaro algorithm computes the similarity between two strings.
4. http://www.altova.com/xml-editor/.
5. Dublin Core Metadata Initiative: http://dublincore.org/.
6. A minimal hierarchy of a dimension *d* is restricted to two parameters: the identifier of *d* directly linked to *All*.

References

Aouabed H., Ben Messaoud, I., Feki, J., and Zurfluh, G. (2012). USD: Un outil d'unification des structures des documents XML. In N. Benblidia & S. Oukidkhouas (Eds.), *Sixième Atelier sur les Systèmes Décisionnels (ASD'12)* (pp. 83–94) Blida, Algeria.

Ben-Abdallah, H., Feki, J., and Ben Abdallah, M., (2009). A multidimensional pattern based approach for the design of data marts. In D. Taniar (Ed.), *Progressive methods in data warehousing and business intelligence: Concepts and competitive analytics, Volume 3 of the Advances in Data Warehousing and Mining Series* (pp. 172–192). Australia: IGI Global.

Ben Messaoud, I., Feki, J., Khrouf, K., and Zurfluh, G. (2011a). "Unification of XML document structures for Document Warehouse (DocW). In *13th International Conference on Enterprise Information Systems (ICEIS'11)* (pp. 85–94). Beijing, China.

Ben Messaoud, I., Feki, J., and Zurfluh, G. (2011b). Modélisation multidimensionnelle des documents XML. *Revue des Nouvelles Technologies de l'Information (RNTI)* B-7, 55–70.

Ben Messaoud, I., Feki, J., and Zurfluh, G. (2012). A first step for building a document warehouse: Unification of XML documents. In *Sixth International Conference on Research Challenges in Information Science (RCIS'12)* (pp. 59–64). Valencia, Spain.

Carpani, F., and Ruggia, R. (2001). An integrity constraints language for a conceptual multidimensional data model. In *13th International Conference on Software Engineering & Knowledge Engineering (SEKE'01)* (pp. 220–227). Argentina.

De-Meo, P., Quattrone, G., Terracina, G., and Ursino, D. (2003). 'Almost automatic' and semantic integration of XML Schemas at various 'severity" levels'.In *Procedings of the International Conference on Cooperative Information Systems (CoopIS)* (pp. 4–21).

Feki, J. (2004). Vers une conception automatisé des entrepôts de données: Modélisation des besoins OLAP et génération de schémas multidimensionnels. In *8th Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI'04)* (pp. 473–485). Sousse, Tunisia: CPU (Centre de Publication Universitaire).

Ghozzi, F., Ravat, F., Teste, O., and Zurfluh, G. (2003). Constraints and multidimensional databases. In *5th International Conference on Enterprise Information Systems (ICEIS'03)* (pp. 104–111). Angers, France.

Hachaichi, Y., Feki, J., & Ben-Abdallah, H. (2010). Modélisation multidimensionnelle de documents XML centrés-données. *Journal of Decision Systems, 19*, 313–345.

Hurtado, C. A., and Mendelzon, A. O. (2002). OLAP Dimension Constraints. In *21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)* (pp. 169–179). Madison, USA.

Inmon, W. H. (2002). *Building the data warehouse*. New York, John Wiley & Sons.

Jaro, M. A. (1989). Advances in record linking methodology as applied to the 1985 census of Tampa Florida. *Journal of the American Statistical Society, 84*, 414–420.

Júnior, C. A. S., and Mello, R. S. (2008). An ontology-driven process for unification of XML instances. In *Brazilian Symposium on Multimedia and the Web* (pp. 242–249). Vila Velha, Brazil.

Kimball, R. (1997). *The data warehouse toolkit*. New York, NY: John Wiley and Sons.

Lee, M. L., Yang, L. H., Hsu, W., and Yang, X. (2002). XClust: Clustering XML schemas for effective integration. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'02)* (pp. 292–299). Virginia, USA.

McCabe, M. C., Lee, J., Chowdhury, A., Grossman, D., and Frieder, O. (2000). On the design and evaluation of a multi-dimensional approach to information retrieval. In Nicholas J. Belkin, Peter Ingwersen, Mun-Kew Leong (Eds.), *Proceedings of the 23th Annual International ACM SIGIR Conference* (pp. 363–365). Athens, Greece.

Mello, R. D. S., Castano, S., & Heuser, C. A. (2002). A method for the unification of XML schemata. *Information and Software Technology, 44*, 241–249.

Pujolle, G., Ravat, F., Teste, O., and Tournier, R. (2011). Multidimensional database design from document-centric XML documents. In Alfredo Cuzzocrea & Umeshwar Dayal (Eds.), *International Conference on Data Warehousing and Knowledge Discovery (DaWaK'11)* (pp. 51–65). Toulouse, France.

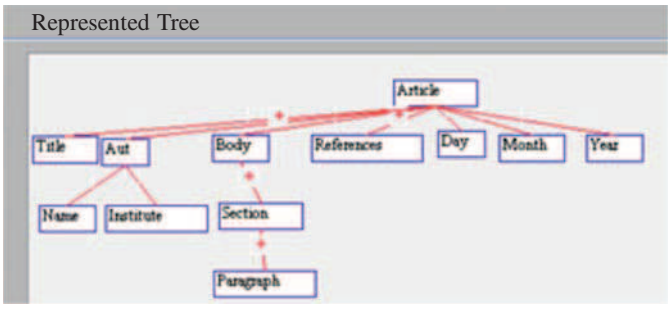
Ravat, F., and Teste, O. (2000). A temporal object-oriented data warehouse model. In *DEXA 2000* (pp. 583–592), London: Springer-Verlag.

Sullivan, D. (2001). *Document warehousing and text mining: Techniques for improving business operations, marketing and sales*. New York, NY: John Wiley & Sons.

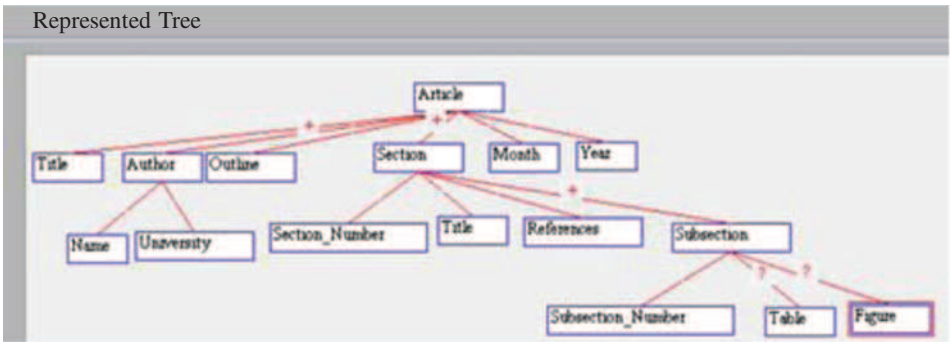
Tseng, F. S. C., & Chou, A. Y. H. (2006). The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. *Decision Support Systems (DSS), 42*, 727–744.
Yoo, C. S., Woo, S. M., and Kim, Y. S. (2005). Unification of XML DTD for XML documents with similar structure. *Computational Science and its Applications – ICCSA,* Part III (pp. 954–963).
Zhang, Y. F., and Liu, W. Y. (2002). Semantic integration of XML schema. *First International Conference on Machine Learning and Cybernetics* (pp. 1085–1061) Beijing, China.

Appendices

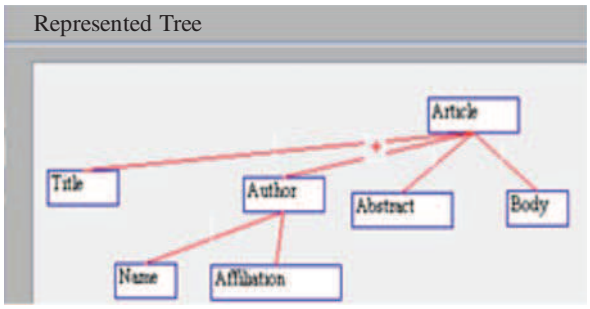
Appendix 1. Interfaces of the *USD* tool.



Tree2: Graphical representation for tree of DTD2.



Tree3: Graphical representation for tree of DTD3.



Tree4: Graphical representation for tree of DTD4.

Appendix 2. Extracted dimensions from the unified tree *Tree-E*.

Dimension				
		Extraction Rule		
Dimension name	**Source node**	*Rd1*	*Rd2*	*Rd3*
D-Article	Article	✓	✓	
D-Author	Author		✓	
D-Date	Day, Month, Year			✓
D-References	References		✓	

Appendix 3. Parameters and weak attributes extracted from *Tree-E*.

		Parameter						Weak attribute			
				Extraction rule						Extraction rule	
Dimension name	Hierarchy Name	Name	Source node	*Rp1*	*Rp2*	*Rp3*	*Rp4*	Name	Source node	*Rw1*	*Rw2*
D-Article	H-Paragraph	Id-D-Article						W-Article-Title	Article-Title	✓	
								W-Abstract	Abstract	✓	
		P-Paragraph	Paragraph		✓						
		P-Section	Section			✓					
		P-Body	Body				✓				
	H-Section-Number	Id-D-Article						W-Article-Title	Article-Title	✓	
								W-Abstract	Abstract	✓	
		P-Section-Number	Section-Number		✓						
		P-Body	Body				✓				
	H-References	Id-D-Article						W-Article-Title	Article-Title	✓	
								W-Abstract	Abstract	✓	
		P-References	References		✓						
		P-Section	Section				✓	W-Section-Title	Section-Title		✓
								W-Section-Number	Section-Number		✓
	H-Subsection	Id-D-Article						W-Article-Title	Article-Title	✓	
								W-Abstract	Abstract	✓	
		P-Subsection	Subsection			✓		W-Table	Table		✓
								W-Figure	Figure		✓
								W-Subsection-Number	Subsection-Number		✓
		P-Section	Section				✓	W-Section-Number	Section-Number		✓
								W-Section-Number	Section-Number		✓
D-Author	H-University	Id-D-Author						W-Name	Name	✓	
		P-University	University	✓							
	H-Affiliation	Id-D-Author						W-Name	Name	✓	
		P-Affiliation	Affiliation	✓							
	H-Institute	Id-D-Author						W-Name	Name	✓	
		P-Institute	Institute	✓							
D-Date	H-Day	Id-D-Date		*Standard Date dimension*							
		P-Day	Day								
		P-Month	Month								
		P-Year	Year								
D-References	H-References	Id-D-References									

Appendix 4. The three DTDs of the medical collection Clef-2007

DTD1

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!ELEMENT WEBURL (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Sex (#PCDATA)>
<!ELEMENT Reviewer (#PCDATA)>
<!ELEMENT References (#PCDATA)>
<!ELEMENT Order (#PCDATA)>
<!ELEMENT OPolytrauma (#PCDATA)>
<!ELEMENT OPathologic (#PCDATA)>
<!ELEMENT OOperation (#PCDATA)>
<!ELEMENT OOpen (#PCDATA)>
<!ELEMENT OLocation (#PCDATA)>
<!ELEMENT OJoint (#PCDATA)>
<!ELEMENT OImplant (#PCDATA)>
<!ELEMENT OGraft (#PCDATA)>
<!ELEMENT ODislocation (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT KeyWords (#PCDATA)>
<!ELEMENT ImageThumbnaillID (#PCDATA)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Hospital (#PCDATA)>
<!ELEMENT Diagnosis (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Department (#PCDATA)>
<!ELEMENT DateTime (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Creation (#PCDATA)>
<!ELEMENT Commentary (#PCDATA)>
<!ELEMENT ClinicalPresentation (#PCDATA)>
<!ELEMENT Chapter (#PCDATA)>
<!ELEMENT CaseID (#PCDATA)>
<!ELEMENT CASIMAGE_CASE ((ID, Description, Diagnosis, Sex, CaseID, ClinicalPresentation,
Commentary, KeyWords, Anatomy, Chapter, ACR, References, Author, Reviewer, Hospital,
Department, State, Date, Language, Title, Birthdate, Age, ImageThumbnaillID, Creation, DateTime,
Order, OJoint, OLocation, OImplant, ODislocation, OPolytrauma, OOpen, OPathologic, OOperation,
OGraft, WEBURL))>
<!ELEMENT Birthdate (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Anatomy (#PCDATA)>
<!ELEMENT Age (#PCDATA)>
<!ELEMENT ACR (#PCDATA)>
```

DTD2

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!ELEMENT WEBURL (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Sex (#PCDATA)>
<!ELEMENT Reviewer (#PCDATA)>
<!ELEMENT References (#PCDATA)>
<!ELEMENT QUESTION (#PCDATA)>
<!ELEMENT QCM ((QUESTION, ANSWERA, ANSWERB, ANSWERC, ANSWERD,
   COMMENTARY))>
```

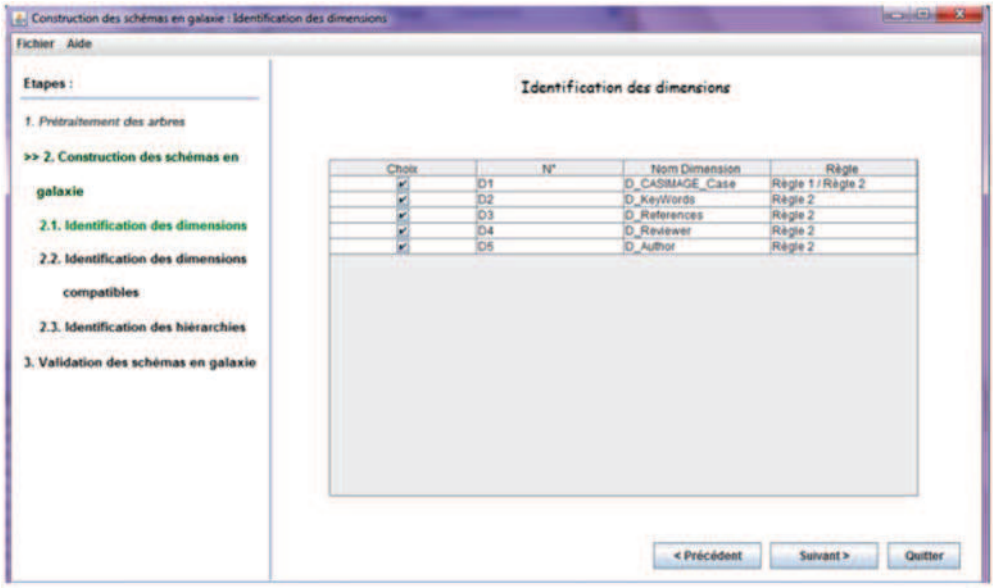
<!ELEMENT Order (#PCDATA)>
<!ELEMENT OPolytrauma (#PCDATA)>
<!ELEMENT OPathologic (#PCDATA)>
<!ELEMENT OOperation (#PCDATA)>
<!ELEMENT OOpen (#PCDATA)>
<!ELEMENT OLocation (#PCDATA)>
<!ELEMENT OJoint (#PCDATA)>
<!ELEMENT OImplant (#PCDATA)>
<!ELEMENT OGraft (#PCDATA)>
<!ELEMENT ODislocation (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT KeyWords (#PCDATA)>
<!ELEMENT ImageThumbnaillID (#PCDATA)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Hospital (#PCDATA)>
<!ELEMENT Diagnosis (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Department (#PCDATA)>
<!ELEMENT DateTime (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Creation (#PCDATA)>
<!ELEMENT Commentary (#PCDATA)>
<!ELEMENT ClinicalPresentation (#PCDATA)>
<!ELEMENT Chapter (#PCDATA)>
<!ELEMENT CaseID (#PCDATA)>
<!ELEMENT COMMENTARY (#PCDATA)>
<!ELEMENT CASIMAGE_CASE ((ID, Description, Diagnosis, Sex, CaseID, ClinicalPresentation+, Commentary, KeyWords+, Anatomy, Chapter, ACR, References+, Author+, Reviewer+, Hospital, Department, State, Date, Language, Title, Birthdate, Age, ImageThumbnaillID, Creation, DateTime, Order, OJoint, OLocation, OImplant, ODislocation, OPolytrauma, OOpen, OPathologic, OOperation, OGraft, QCM+, WEBURL))>
<!ELEMENT Birthdate (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Anatomy (#PCDATA)>
<!ELEMENT Age (#PCDATA)>
<!ELEMENT ANSWERD (#PCDATA)>
<!ELEMENT ANSWERC (#PCDATA)>
<!ELEMENT ANSWERB (#PCDATA)>
<!ELEMENT ANSWERA (#PCDATA)>
<!ELEMENT ACR (#PCDATA)>

DTD3

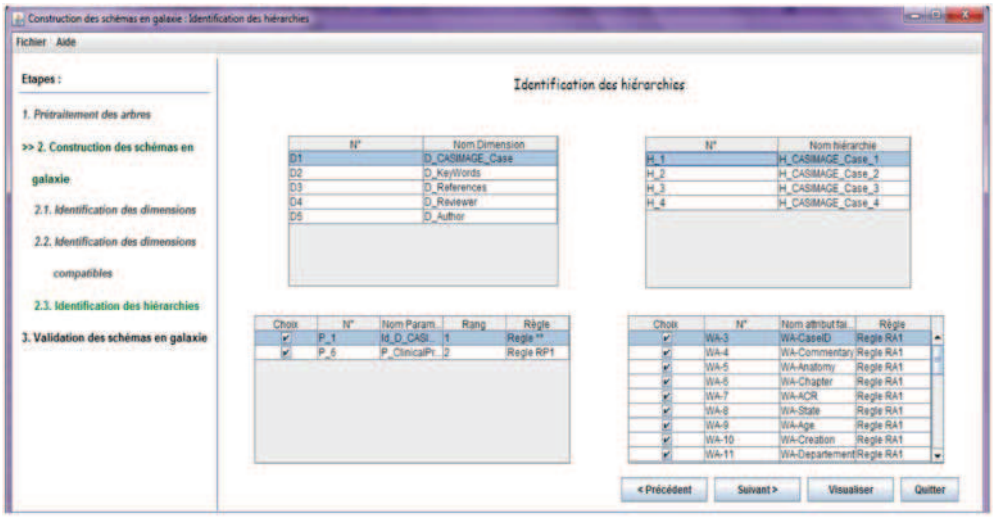
<?xml version=”1.0” encoding=”UTF-8”?>
<!ELEMENT WEBURL (#PCDATA)>
<!ELEMENT WEBLINK ((URL, DESCRIPTION))>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT State EMPTY>
<!ELEMENT Sex EMPTY>
<!ELEMENT Reviewer EMPTY>
<!ELEMENT References EMPTY>
<!ELEMENT QUESTION (#PCDATA)>
<!ELEMENT QCM ((QUESTION, ANSWERA, ANSWERB, ANSWERC, ANSWERD, COMMENTARY))>
<!ELEMENT Order (#PCDATA)>
<!ELEMENT OPolytrauma (#PCDATA)>
<!ELEMENT OPathologic (#PCDATA)>

<!ELEMENT OOperation (#PCDATA)>
<!ELEMENT OOpen (#PCDATA)>
<!ELEMENT OLocation EMPTY>
<!ELEMENT OJoint EMPTY>
<!ELEMENT OImplant EMPTY>
<!ELEMENT OGraft (#PCDATA)>
<!ELEMENT ODislocation (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT LINK ((ID, COMMENTARY))>
<!ELEMENT KeyWords EMPTY>
<!ELEMENT ImageThumbnaillID (#PCDATA)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Hospital (#PCDATA)>
<!ELEMENT Diagnosis (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Department (#PCDATA)>
<!ELEMENT DateTime (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT Creation (#PCDATA)>
<!ELEMENT Commentary EMPTY>
<!ELEMENT ClinicalPresentation (#PCDATA)>
<!ELEMENT Chapter (#PCDATA)>
<!ELEMENT CaseID EMPTY>
<!ELEMENT COMMENTARY (#PCDATA)>
<!ELEMENT CASIMAGE_CASE ((ID, Description, Diagnosis, Sex, CaseID, ClinicalPresentation, Commentary, KeyWords, Anatomy, Chapter, ACR, References, Author, Reviewer, Hospital, Department, State, Date, Language, Title, Birthdate, Age, ImageThumbnaillID, Creation, DateTime, Order, OJoint, OLocation, OImplant, ODislocation, OPolytrauma, OOpen, OPathologic, OOperation, OGraft, QCM, LINK+, WEBLINK+, WEBURL))>
<!ELEMENT Birthdate (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Anatomy (#PCDATA)>
<!ELEMENT Age (#PCDATA)>
<!ELEMENT ANSWERD (#PCDATA)>
<!ELEMENT ANSWERC (#PCDATA)>
<!ELEMENT ANSWERB (#PCDATA)>
<!ELEMENT ANSWERA (#PCDATA)>
<!ELEMENT ACR EMPTY>

Appendix 5. Interfaces of the *Galaxy-Gen* tool.



Interface of extracted dimensions for the galaxy: Clef galaxy



Extracted hierarchies for the D_Casimage_Case dimension